```
//
//
//     ░██   ██░▓█████▄ ▓█████  ▄████▄   ░█████   █     █░
//     ▒▒ █ █ ▒░▒██▀ ██▌▓█   ▀ ▒██▀ ▀█  ▒██▒  ██▒▓█░ █ ░█░
//     ░░  █   ░░██   █▌▒███   ▒▓█    ▄ ▒██░  ██▒▒█░ █ ░█
//      ░ █ █ ▒ ░▓█▄   ▌▒▓█  ▄ ▒▓▓▄ ▄██▒▒██   ██░░█░ █ ░█
//     ▒██▒ ▒██▒░▒████▓ ░▒████▒▒ ▓███▀ ░░ ████▓▒░░░██▒██▓
//     ▒▒ ░ ░▓ ░ ▒▒▓  ▒ ░░ ▒░ ░░ ░▒ ▒  ░░ ▒░▒░▒░ ░ ▓░▒ ▒
//     ░░   ░▒ ░ ░ ▒  ▒  ░ ░  ░  ░  ▒     ░ ▒ ▒░   ▒ ░ ░
//      ░    ░   ░ ░  ░    ░   ░        ░ ░ ░ ▒    ░   ░
//      ░    ░     ░       ░  ░░ ░          ░ ░      ░
//           ░             ░
//
// This source code is subject to the terms of the Mozilla Public License 2.0 at https://mozilla.org/M
// © xdecow

//@version=5
indicator('Multiple MTF Moving Average [xdecow]', shorttitle='Multiple MTF MA [xdecow]', overlay=true)




//------------------------------------------------------------------------------
// Inputs

// global input
i_hideFromLowerTf = input.bool(true, 'Hide MA from lower timeframe')
i_mtf_mode = input.bool(true, "MTF Smoothed Mode", tooltip = "Smoothes the average between the bars o

// label input
const string g_label = 'MA Labels'
i_showType = input.bool(true, 'Display MA Type', group = g_label)
i_showLen = input.bool(true, 'Display MA Length', group = g_label)
i_showTf = input.bool(true, 'Display MA Timeframe', group = g_label)
i_showN = input.bool(false, 'Display MA Number', group = g_label, tooltip = 'Helps you find and confi
i_labelOffset = input.int(0, 'Label offset', minval=0, group = g_label)
i_labelSize = input.string('normal', 'Label size', options = ['tiny', 'small', 'normal', 'large', 'hug


//------------------------------------------------------------------------------
// Funcs


f_mvwap(src, length, vol) =>
    uvol = vol
    uwgt = vol * src

    if length < 1
        na
    else
        for i = 1 to length - 1
            uvol := uvol + vol[i]
            uwgt := uwgt + vol[i] * src[i]
        uwgt / uvol


f_dema(_src, _length) =>
    e1 = ta.ema(_src, _length)
    e2 = ta.ema(e1, _length)
    2 * e1 - e2


f_tema(_src, _length) =>
    e1 = ta.ema(_src, _length)
    e2 = ta.ema(e1, _length)
    e3 = ta.ema(e2, _length)
    (3 * e1) - (3 * e2) + e3


f_smma(_src, _length) =>
    sma = ta.sma(_src, _length)
    float smma = 0.0
    smma := na(smma[1]) ? sma : (smma[1] * (_length - 1) + _src) / _length


f_ma(_type, _src, _len) =>
    switch _type
        "EMA: Exponential Moving Average" => ta.ema(_src, _len)
        "DEMA: Double Exponential Moving Average" => f_dema(_src, _len)
        "TEMA: Triple Exponential Moving Average" => f_tema(_src, _len)
        "VWMA: Volume Weighted Moving Average" => ta.vwma(_src, _len)
        "RMA: Rolling Moving Average" => ta.rma(_src, _len)
        "WMA: Weighted Moving Average" => ta.wma(_src, _len)
        "HMA: Hull Moving Average" => ta.hma(_src, _len)
        "SMA: Simple Moving Average" => ta.sma(_src, _len)
        "SWMA: Symmetrically Weighted Moving Average" => ta.swma(_src)
        "SMMA: Smoothed Moving Average" => f_smma(_src, _len)
        "VWAP: Volume Weighted Average Price (Daily)" => ta.vwap(_src)
        "MVWAP: Moving VWAP" => f_mvwap(_src, _len, volume)
        "LSMA: Least Squares Moving Average" => ta.linreg(_src, _len, 0)


f_timeframeToHuman(_tf) =>
    seconds = timeframe.in_seconds(_tf)

    if seconds < 60
        _tf
    else if seconds < 3600
        str.tostring(seconds / 60) + 'm'
    else if seconds < 86400
        str.tostring(seconds / 60 / 60) + 'h'
    else
        switch _tf
            "1D" => "D"
            "1W" => "W"
            "1M" => "M"
            => str.tostring(_tf)
```

```
f_getSizeFromString(_size) =>
    switch _size
        "auto" => size.auto
        "tiny" => size.tiny
        "small" => size.small
        "normal" => size.normal
        "large" => size.large
        "huge" => size.huge


f_getLabelText(_tf, _type, _len, _n) =>
    t = ''
    maType = str.substring(_type, 0, str.pos(_type, ':'))

    if i_showType
        t := maType

    if i_showLen and maType != "VWAP" and maType != "SWMA"
        t += (t != '' ? ' ' : '') + str.tostring(_len)

    if i_showTf and _tf != ''
        t += (t != '' ? ' / ' : '') + f_timeframeToHuman(_tf)

    if i_showN
        t += (t != '' ? ' ' : '') + '#' + str.tostring(_n)
    t


labelSize = f_getSizeFromString(i_labelSize)

f_addMaLabel(_enabled, _n, _src, _price, _tf, _type, _len, _color, _offset) =>
    label l = na
    if _enabled and not na(_price)
        labelText = f_getLabelText(_tf, _type, _len, _n)
        if labelText != ''
            pos = chart.point.from_index(bar_index + i_labelOffset + _offset, _price)
            maTooltip = str.format("MA #{0}\nType: {1}\nLength: {2}\nTimeframe: {3}\nPrice: {4}", _n,
            l := label.new(pos, text=labelText, color=color.new(color.black, 100), textcolor=_color, s
    label.delete(l[1])



// real value / step lines
f_getMaConverted_mode2(_tf, _type, _src, _len) =>
    ma = f_ma(_type, _src, _len)
    tfma = request.security(syminfo.tickerid, _tf, ma)

// smoothed lines
f_getMaConverted_mode1(_tf, _type, _src, _len) =>
    ma = f_ma(_type, _src, _len)
    [m, bindex] = request.security(syminfo.tickerid, _tf, [ma, bar_index])
    int lbindex = na
    lbindex := na(lbindex[1]) or bindex > lbindex[1] ? bindex : lbindex[1]
    lbindex != lbindex[1] or barstate.islast ? m : na

f_getMaConverted(_tf, _type, _src, _len) =>
    if i_mtf_mode
        f_getMaConverted_mode1(_tf, _type, _src, _len)
    else
        f_getMaConverted_mode2(_tf, _type, _src, _len)



f_showInCurrentTimeframe(_hideInLowerTf, _tf) =>
    if _hideInLowerTf == false
        true
    else
        timeframe.in_seconds(_tf) >= timeframe.in_seconds()




//------------------------------------------------------------------------------
// Plots


// line width
const int C_LINEWIDTH = 2


//------------------------------------
// MA #1
const string g_ma1 = 'Moving Average #1'
i_ma1_enabled = input.bool(true, 'Enabled', group = g_ma1)
i_ma1_tf = input.timeframe('', 'Timeframe', group = g_ma1)
i_ma1_src = input.source(close, 'Source', group = g_ma1)
i_ma1_len = input.int(10, 'Length', group = g_ma1)
i_ma1_offset = input.int(0, 'Offset', group = g_ma1)
i_ma1_type = input.string('EMA: Exponential Moving Average', 'Type', options=["DEMA: Double Exponentia
i_ma1_color = input.color(#f7525f, 'Color', group = g_ma1)

v_ma1 = f_getMaConverted(i_ma1_tf, i_ma1_type, i_ma1_src, i_ma1_len)
v_ma1_visible = i_ma1_enabled and f_showInCurrentTimeframe(i_hideFromLowerTf, i_ma1_tf)
plot(v_ma1_visible ? v_ma1 : na, color=i_ma1_color, linewidth=C_LINEWIDTH, title='MA #1', join=true, c
f_addMaLabel(v_ma1_visible, 1, i_ma1_src, v_ma1, i_ma1_tf, i_ma1_type, i_ma1_len, i_ma1_color, i_ma1_c


//------------------------------------
// MA #2
const string g_ma2 = 'Moving Average #2'
i_ma2_enabled = input.bool(true, 'Enabled', group = g_ma2)
i_ma2_tf = input.timeframe('', 'Timeframe', group = g_ma2)
i_ma2_src = input.source(close, 'Source', group = g_ma2)
i_ma2_len = input.int(20, 'Length', group = g_ma2)
i_ma2_offset = input.int(0, 'Offset', group = g_ma2)
i_ma2_type = input.string('EMA: Exponential Moving Average', 'Type', options=["DEMA: Double Exponentia
i_ma2_color = input.color(#ffa726, 'Color', group = g_ma2)
```

```
v_ma2 = f_getMaConverted(i_ma2_tf, i_ma2_type, i_ma2_src, i_ma2_len)
v_ma2_visible = i_ma2_enabled and f_showInCurrentTimeframe(i_hideFromLowerTf, i_ma2_tf)
plot(v_ma2_visible ? v_ma2 : na, color=i_ma2_color, linewidth=C_LINEWIDTH, title='MA #2', join=true, c
f_addMaLabel(v_ma2_visible, 2, i_ma2_src, v_ma2, i_ma2_tf, i_ma2_type, i_ma2_len, i_ma2_color, i_ma2_c


//------------------------------------
// MA #3
const string g_ma3 = 'Moving Average #3'
i_ma3_enabled = input.bool(true, 'Enabled', group = g_ma3)
i_ma3_tf = input.timeframe('', 'Timeframe', group = g_ma3)
i_ma3_src = input.source(close, 'Source', group = g_ma3)
i_ma3_len = input.int(30, 'Length', group = g_ma3)
i_ma3_offset = input.int(0, 'Offset', group = g_ma3)
i_ma3_type = input.string('EMA: Exponential Moving Average', 'Type', options=["DEMA: Double Exponentia
i_ma3_color = input.color(#ffee58, 'Color', group = g_ma3)

v_ma3 = f_getMaConverted(i_ma3_tf, i_ma3_type, i_ma3_src, i_ma3_len)
v_ma3_visible = i_ma3_enabled and f_showInCurrentTimeframe(i_hideFromLowerTf, i_ma3_tf)
plot(v_ma3_visible ? v_ma3 : na, color=i_ma3_color, linewidth=C_LINEWIDTH, title='MA #3', join=true, c
f_addMaLabel(v_ma3_visible, 3, i_ma3_src, v_ma3, i_ma3_tf, i_ma3_type, i_ma3_len, i_ma3_color, i_ma3_c


//------------------------------------
// MA #4
const string g_ma4 = 'Moving Average #4'
i_ma4_enabled = input.bool(true, 'Enabled', group = g_ma4)
i_ma4_tf = input.timeframe('', 'Timeframe', group = g_ma4)
i_ma4_src = input.source(close, 'Source', group = g_ma4)
i_ma4_len = input.int(40, 'Length', group = g_ma4)
i_ma4_offset = input.int(0, 'Offset', group = g_ma4)
i_ma4_type = input.string('EMA: Exponential Moving Average', 'Type', options=["DEMA: Double Exponentia
i_ma4_color = input.color(#66bb6a, 'Color', group = g_ma4)

v_ma4 = f_getMaConverted(i_ma4_tf, i_ma4_type, i_ma4_src, i_ma4_len)
v_ma4_visible = i_ma4_enabled and f_showInCurrentTimeframe(i_hideFromLowerTf, i_ma4_tf)
plot(v_ma4_visible ? v_ma4 : na, color=i_ma4_color, linewidth=C_LINEWIDTH, title='MA #4', join=true, c
f_addMaLabel(v_ma4_visible, 4, i_ma4_src, v_ma4, i_ma4_tf, i_ma4_type, i_ma4_len, i_ma4_color, i_ma4_c


//------------------------------------
// MA #5
const string g_ma5 = 'Moving Average #5'
i_ma5_enabled = input.bool(true, 'Enabled', group = g_ma5)
i_ma5_tf = input.timeframe('', 'Timeframe', group = g_ma5)
i_ma5_src = input.source(close, 'Source', group = g_ma5)
i_ma5_len = input.int(50, 'Length', group = g_ma5)
i_ma5_offset = input.int(0, 'Offset', group = g_ma5)
i_ma5_type = input.string('EMA: Exponential Moving Average', 'Type', options=["DEMA: Double Exponentia
i_ma5_color = input.color(#22ab94, 'Color', group = g_ma5)

v_ma5 = f_getMaConverted(i_ma5_tf, i_ma5_type, i_ma5_src, i_ma5_len)
v_ma5_visible = i_ma5_enabled and f_showInCurrentTimeframe(i_hideFromLowerTf, i_ma5_tf)
plot(v_ma5_visible ? v_ma5 : na, color=i_ma5_color, linewidth=C_LINEWIDTH, title='MA #5', join=true, c
f_addMaLabel(v_ma5_visible, 5, i_ma5_src, v_ma5, i_ma5_tf, i_ma5_type, i_ma5_len, i_ma5_color, i_ma5_c
//-----------------------------------
// MA #6
const string g_ma6 = 'Moving Average #6'
i_ma6_enabled = input.bool(true, 'Enabled', group = g_ma6)
i_ma6_tf = input.timeframe('', 'Timeframe', group = g_ma6)
i_ma6_src = input.source(close, 'Source', group = g_ma6)
i_ma6_len = input.int(60, 'Length', group = g_ma6)
i_ma6_offset = input.int(0, 'Offset', group = g_ma6)
i_ma6_type = input.string('EMA: Exponential Moving Average', 'Type', options=["DEMA: Double Exponentia
i_ma6_color = input.color(#26c6da, 'Color', group = g_ma6)

v_ma6 = f_getMaConverted(i_ma6_tf, i_ma6_type, i_ma6_src, i_ma6_len)
v_ma6_visible = i_ma6_enabled and f_showInCurrentTimeframe(i_hideFromLowerTf, i_ma6_tf)
plot(v_ma6_visible ? v_ma6 : na, color=i_ma6_color, linewidth=C_LINEWIDTH, title='MA #6', join=true, c
f_addMaLabel(v_ma6_visible, 6, i_ma6_src, v_ma6, i_ma6_tf, i_ma6_type, i_ma6_len, i_ma6_color, i_ma6_c


//-----------------------------------
// MA #7
const string g_ma7 = 'Moving Average #7'
i_ma7_enabled = input.bool(true, 'Enabled', group = g_ma7)
i_ma7_tf = input.timeframe('', 'Timeframe', group = g_ma7)
i_ma7_src = input.source(close, 'Source', group = g_ma7)
i_ma7_len = input.int(70, 'Length', group = g_ma7)
i_ma7_offset = input.int(0, 'Offset', group = g_ma7)
i_ma7_type = input.string('EMA: Exponential Moving Average', 'Type', options=["DEMA: Double Exponentia
i_ma7_color = input.color(#3179f5, 'Color', group = g_ma7)

v_ma7 = f_getMaConverted(i_ma7_tf, i_ma7_type, i_ma7_src, i_ma7_len)
v_ma7_visible = i_ma7_enabled and f_showInCurrentTimeframe(i_hideFromLowerTf, i_ma7_tf)
plot(v_ma7_visible ? v_ma7 : na, color=i_ma7_color, linewidth=C_LINEWIDTH, title='MA #7', join=true, c
f_addMaLabel(v_ma7_visible, 7, i_ma7_src, v_ma7, i_ma7_tf, i_ma7_type, i_ma7_len, i_ma7_color, i_ma7_c


//-----------------------------------
// MA #8
const string g_ma8 = 'Moving Average #8'
i_ma8_enabled = input.bool(true, 'Enabled', group = g_ma8)
i_ma8_tf = input.timeframe('', 'Timeframe', group = g_ma8)
i_ma8_src = input.source(close, 'Source', group = g_ma8)
i_ma8_len = input.int(80, 'Length', group = g_ma8)
i_ma8_offset = input.int(0, 'Offset', group = g_ma8)
i_ma8_type = input.string('EMA: Exponential Moving Average', 'Type', options=["DEMA: Double Exponentia
i_ma8_color = input.color(#7e57c2, 'Color', group = g_ma8)

v_ma8 = f_getMaConverted(i_ma8_tf, i_ma8_type, i_ma8_src, i_ma8_len)
v_ma8_visible = i_ma8_enabled and f_showInCurrentTimeframe(i_hideFromLowerTf, i_ma8_tf)
plot(v_ma8_visible ? v_ma8 : na, color=i_ma8_color, linewidth=C_LINEWIDTH, title='MA #8', join=true, c
f_addMaLabel(v_ma8_visible, 8, i_ma8_src, v_ma8, i_ma8_tf, i_ma8_type, i_ma8_len, i_ma8_color, i_ma8_c


//-----------------------------------
// MA #9
```

```
const string g_ma9 = 'Moving Average #9'
i_ma9_enabled = input.bool(true, 'Enabled', group = g_ma9)
i_ma9_tf = input.timeframe('', 'Timeframe', group = g_ma9)
i_ma9_src = input.source(close, 'Source', group = g_ma9)
i_ma9_len = input.int(90, 'Length', group = g_ma9)
i_ma9_offset = input.int(0, 'Offset', group = g_ma9)
i_ma9_type = input.string('EMA: Exponential Moving Average', 'Type', options=["DEMA: Double Exponentia
i_ma9_color = input.color(#ab47bc, 'Color', group = g_ma9)

v_ma9 = f_getMaConverted(i_ma9_tf, i_ma9_type, i_ma9_src, i_ma9_len)
v_ma9_visible = i_ma9_enabled and f_showInCurrentTimeframe(i_hideFromLowerTf, i_ma9_tf)
plot(v_ma9_visible ? v_ma9 : na, color=i_ma9_color, linewidth=C_LINEWIDTH, title='MA #9', join=true, c
f_addMaLabel(v_ma9_visible, 9, i_ma9_src, v_ma9, i_ma9_tf, i_ma9_type, i_ma9_len, i_ma9_color, i_ma9_c


//------------------------------------
// MA #10
const string g_ma10 = 'Moving Average #10'
i_ma10_enabled = input.bool(true, 'Enabled', group = g_ma10)
i_ma10_tf = input.timeframe('', 'Timeframe', group = g_ma10)
i_ma10_src = input.source(close, 'Source', group = g_ma10)
i_ma10_len = input.int(100, 'Length', group = g_ma10)
i_ma10_offset = input.int(0, 'Offset', group = g_ma10)
i_ma10_type = input.string('EMA: Exponential Moving Average', 'Type', options=["DEMA: Double Exponenti
i_ma10_color = input.color(#ec407a, 'Color', group = g_ma10)

v_ma10 = f_getMaConverted(i_ma10_tf, i_ma10_type, i_ma10_src, i_ma10_len)
v_ma10_visible = i_ma10_enabled and f_showInCurrentTimeframe(i_hideFromLowerTf, i_ma10_tf)
plot(v_ma10_visible ? v_ma10 : na, color=i_ma10_color, linewidth=C_LINEWIDTH, title='MA #10', join=tru
f_addMaLabel(v_ma10_visible, 10, i_ma10_src, v_ma10, i_ma10_tf, i_ma10_type, i_ma10_len, i_ma10_color,
```

PDF document made with CodePrint using Prism